High-performance Data Acquisition and Beamline Control: Current Capabilities and Future Needs

Mark Rivers

GeoSoilEnviroCARS

Advanced Photon Source

University of Chicago

Outline

- Needs for high performance beamline control and data acquisition
- Where are we now
  – Strengths
  – Weaknesses
- What needs to be done for the future

Characteristics of an Ideal Beamline Control and Data Acquisition System

1. Fast, efficient
   - Data collection time limited only by photon count rate on detector
   - Expert users don't want to navigate GUI
2. Easy to use
   - Intuitive Graphical User Interface (GUI)
   - Customized for the specific experiment
3. Powerful, able to accomplish any task, even if it has never been done before
4. Able to use existing applications that have substantial development and user familiarity (GDA, Blu-Ice, SPEC, someones favorite EXAFS program, etc.)
5. Can use different applications to run the same equipment for different experiments (i.e. not tied into Labview or SPEC drivers)

Strengths of synApps/EPICS solution

- Distributed control system is clearly the correct solution now and in the future
- Members of a large collaboration (Diamond, SLS, NSLS-II, AS, SSRF)
- Very flexible
- Modular
- Configuration versus programming
- Can scan anything, read anything

Weaknesses of synApps/EPICS solution

- MX community has it easier, not much changes day-to-day, they have developed good user interfaces
- No good user-interface for the “experiment” beamlines
- Many solutions to the same problem or at least similar problems
- SPEC has been the only tool that understands reciprocal space and diffractometer geometries
- Many GUIs Python, IDL, etc. No unifying higher level tool set (plotting, data files, analysis tools)

asyn
A tool that makes it easy to add new complex devices to EPICS

- Well defined interface between EPICS device support and driver
- Standard asyn device support that can be used in nearly all cases
- In last 8 years I have written a LOT of new drivers and I have written NO device support, just use standard asyn device support
- I believe asyn should be used to write *all* EPICS device drivers, not just "asynchronous" drivers like serial, GPIB and TCP/IP.
  - All of my drivers (except one old one) use asyn
- There is a new C++ base class, asynPortDriver, that makes it very easy to write a new driver
  - All of my areaDetector, D/A, binary I/O, and most recently motor drivers use asynPortDriver

motor module

- Motor record and device support
- Motors are the single most important component of beamline control
- Provides:
  - stepper and servo motors
    - VME, CAMAC, RS-232, GPIB, Ethernet controllers
    - More than 20 types at present
  - soft-motor support
    - Put motor “face” on, e.g., a DAC channel
    - Drive a hard motor through a nonlinear transform of 1 or more real motors
  - user/dial/raw coordinates
  - backlash-takeout algorithm
  - pre/post move commands
  - many more features

motor module
3 types of device/driver support

1. Old, pre-asyn.
   – Separate device and driver support for each controller type.
   – Very hard to add support for features beyond those supported by motor record
2. Newer, asyn-based.
   – Device independent device support, device independent asyn driver.
   – Need to write an asyn-independed driver for a new controller
   – Much better than 1) above, but still hard to add support for controller-specific features
3. Newest: asynMotorDriver C++ class, derived from asynPortDriver
   – Easy to take advantage of controller-specific features
   – Only simulation driver at present
   – I plan to convert Newport XPS and OMS MAXv
   – Can then add complex coordinated motion in a standard manner
   – This is what NSLS-II should use for any new motor drivers

Coordinated multi-axis on-the-fly scanning with EPICS

- Newport XPS (and older MM4005) motor controllers support complex coordinated motion in 8-axis space
- Put out synchronization pulses at specified intervals along the trajectory
  - Used to trigger detectors (e.g. SIS multi-channel scaler, Pilatus, XIA xMAP, etc.) for data collection
- Can read back from the controller the actual positions of every encoder when each pulse was output so actual motor positions are known even if these is a following error.

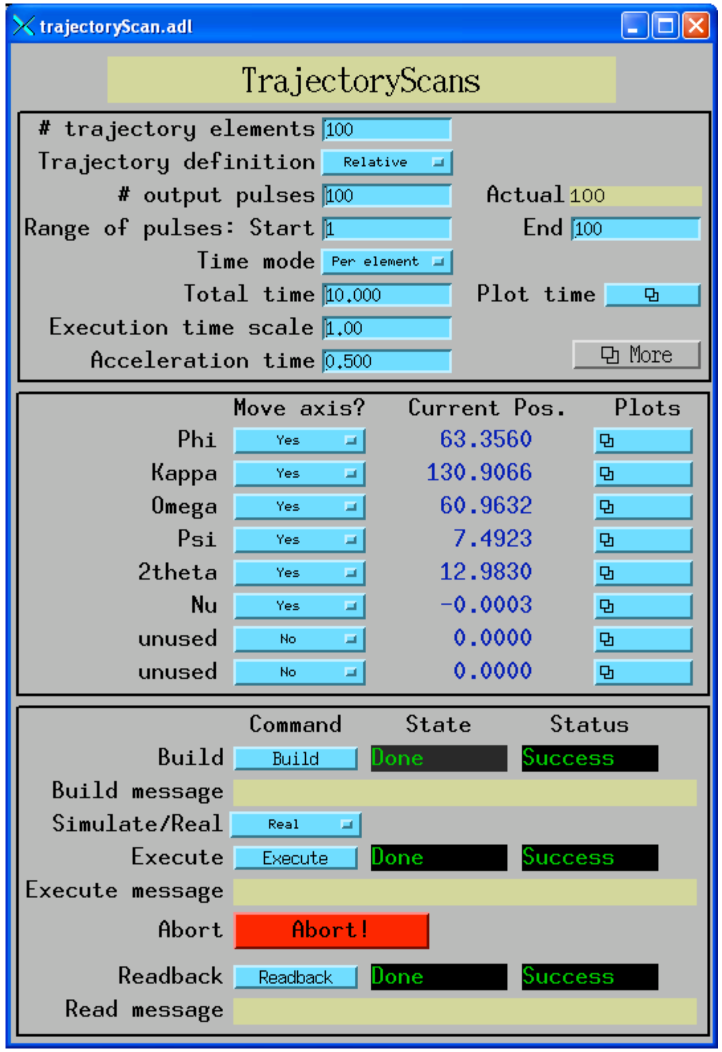
Use with Large Newport Diffractometers

- 5 installed at APS (GSECARS (2), sectors 2, 33, 34)
- Large goniometers have long setting times (>.5 sec) for short moves
- On-axis encoders and XPS permit rapid on-the-fly scanning with <.001 degree position errors
- Many other stages at GSECARS use XPS controllers now, beginning to use XPS to drive x-ray fluorescence maps and other applications.

EPICS Trajectory Scanning Software

- Common database for any controller
- Define the total number of trajectory elements.
  - Controller interpolates between trajectory elements. Simple linear motion thus requires only 2 trajectory elements, no matter how many "points" are being measured.
- Define the absolute or relative position of each axis for each point in the trajectory.
- Define the time for each element of the trajectory, or alternatively the total execution time with equal time per trajectory element.
- Define the total number of output synchronization pulses
- Define the trajectory elements where pulse outputs begin and end.
- Build the trajectory, checking for completion and errors.
- Execute the trajectory, checking for completion and errors. This can be done repeatedly without rebuilding if the only changes are in the start position.
- Read back the actual position of each axis when each synchronization pulse was output.
- All documented at
  - http://cars.uchicago.edu/software/trajectoryScan.html

medm screen
SPEC, IDL or Python client defines actual trajectory



EPICS Implementation

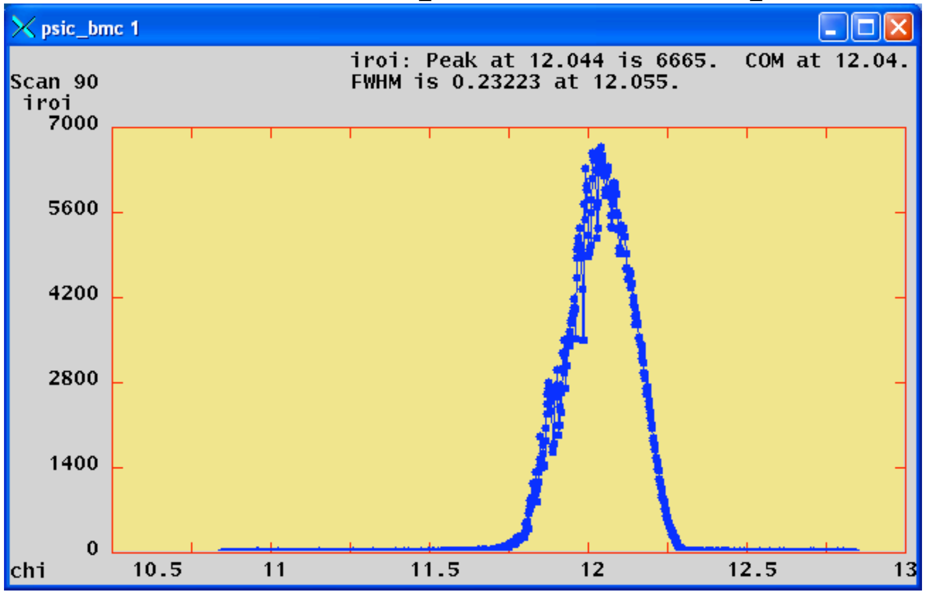
- A database file, *trajectoryScan.db*.
  - Contains almost no "logic" with no links between records in the database. The records are simply variables which channel access clients and the State Notation Language (SNL) program use. Waveform records define motor positions in trajectory
- SNL program, MM4005_*trajectoryScan.st*, XPS_trajectoryScan.st.
  - Implements all of the logic for communicating with the MM4005 and XPS, and with channel access clients via the database.
- MEDM screens, *trajectoryScan.adl, trajectoryScanDebug.adl, trajectoryPlot.adl*.
  - Used to control the building, execution, readback, debugging and plotting of trajectory scans.
- Goes "behind the back" of the motor record driver to talk directly to the controller.
  - Not the right way to do it, it should be a function that motor controllers can optionally support
  - Need to define the API for coordinated motion

SPEC Interface

- SPEC macros have been written to allow SPEC to utilize trajectory scanning via EPICS interface
- Low level interface, all of SPEC's standard scans can be done "on-the-fly" with trajectory scanning software. Replacement macros for:
  - _ascan # Used by all ascan and dscan macros
  - Mesh
  - hklscan # Used by hscan, kscan and lscan
  - _hklmesh
  - _hklline # Used by hkcircle, hlcircle, klcircle, hkradial, hlradial and klradial
  - _scanabort resume
  - _loop
- SPEC is basically being used as a calculator to compute the motor positions at each point in the scan, the actual motion is being done down in the controller
- Enormous improvement in performance, while keeping complete compatibility with SPEC step-scanning
- Can do a 1000-point alignment scan with a point detector or a Pilatus in under 20 seconds. Compare to many minutes step-scanning

Performance Example with Pilatus driver

- SPEC used to collect 1000 points using trajectory scanning mode with the Newport XPS motor controller. Hardware trigger of Pilatus from XPS.
- Relative scan of the chi axis from -2 degrees to +2 degrees with 1000 points at .02 seconds/point
- Coordinated motion of the phi, kappa and omega axes.
- Theoretical time 20.0 second, actual time 20.8 seconds
- Includes time to save all 1000 images to disk (366 MB), Pilatus driver to read each file, correct bad pixels and flat field, compute ROIs, and post the ROIs and 1000 images to EPICS.



Future Plans for Trajectory Scanning Software

- Current architecture is not correct.
- 2 independent paths to motor controller: SNL program and motor record driver
- New architecture will put the coordinated motion down in the motor driver
- Take advantage of capabilities of specific controllers
- Perhaps even emulate capability in dumb controllers
- Test case will be OMS MAXv, my next major project
- NSLS-II is looking seriously at moving beyond the EPICS motor record, putting the state logic elsewhere
  – This needs to be gotten right, there will probably only be 1 chance to do it again for a long time

areaDetector module

- Drivers for many detectors popular at synchrotron beamlines
  - Handle detectors ranging from >500 frames/second to <1 frame/second
- Basic parameters for all detectors
  - E.g. exposure time, start acquisition, etc.
  - Allows generic clients to be used for many applications
- Easy to implement new detector
  - Single device-driver C++ file to write. EPICS independent.
- Easy to implement detector-specific features
  - Driver understands additional parameters beyond those in the basic set
- EPICS-independent at lower layers.
- Middle-level plug-ins to add capability like regions-of-interest calculation, file saving, etc.
  - Device independent, work with all drivers
  - Below the EPICS layer for highest performance

areaDetector plugins

- Designed to perform real-time processing of data, running in the EPICS IOC (not over EPICS Channel Access)
- Receive NDArray data over callbacks from drivers or other plugins
- Plug-ins can execute in their own threads (non-blocking) or in callback thread (blocking)
  - If non-blocking then NDArray data is queued
    - Can drop images if queue is full
  - If executing in callback thread, no queuing, but slows device driver
- Allows
  - Enabling/disabling
  - Throttling rate (no more than 0.5 seconds, etc)
  - Changing data source for NDArray callbacks to another driver or plugin
- Some plugins are also sources of NDArray callbacks, as well as consumers.
  - Allows creating a data processing pipeline running at very high speed, each in a difference thread, and hence in multiple cores on modern CPUs.

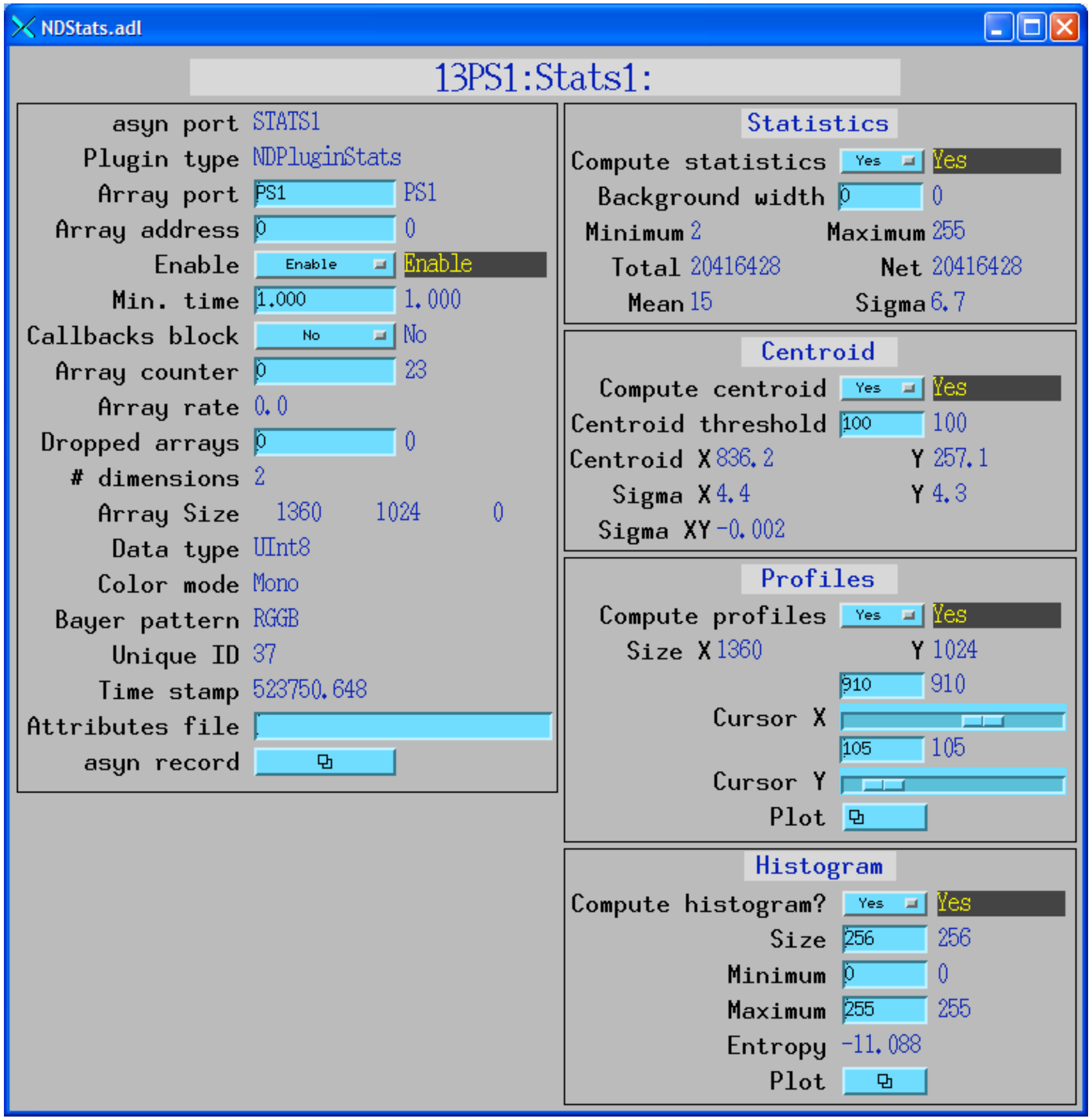
areaDetector plugins

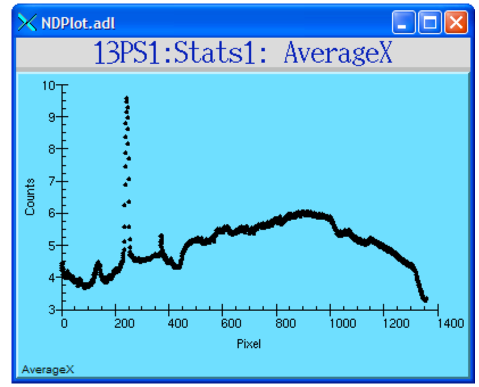
- NDPlugInStdArrays
  - Receives arrays (images) from device drivers, converts to standard arrays, e.g. waveform records.
  - This plugin is what EPICS channel access viewers normally talk to.
- NDPluginROI
  - Performs region-of-interest calculations
  - Select a subregion. Optionally bin, reverse in either direction, convert data type, scale by a constant.
- NDPluginColorConvert
  - Convert from one color model to another (Mono, Bayer, RGB pixel, row or planar interleave)
- NDPluginMJPEG (Diamond)
  - MJPEG server that allows viewing images in a Web browser.

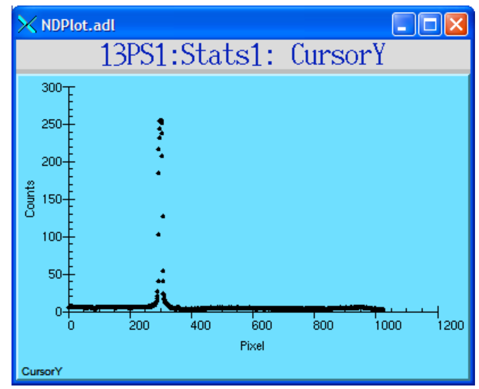
areaDetector plugins (new in R1-6)

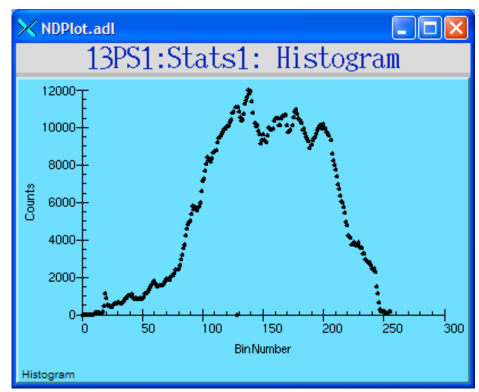
- NDPluginStats
  - Calculates statistics on an array
  - Replaces the statistics calculations that were previously performed in the ROI plugin.
  - Adds new statistics, including the centroid position and width.
  - Computes X and Y profiles, including average profiles, profiles at the centroid position, and profiles at a user-defined cursor position.
- NDPluginProcess
  - Does arithmetic processing on arrays
  - Background subtraction.
  - Flat field normalization.
  - Offset and scale.
  - Low and high clipping.
  - Recursive filtering in the time domain.
  - Conversion to a different output data type.
- NDPluginOverlay
  - Adds graphic overlays to an image.
  - Replaces the "Highligh ROIs" function that was previously provided in the ROI plugin.
  - Much more general, and can be used to display not only ROIs, but multiple cursors, user-defined boxes, etc.

Statistics plugin

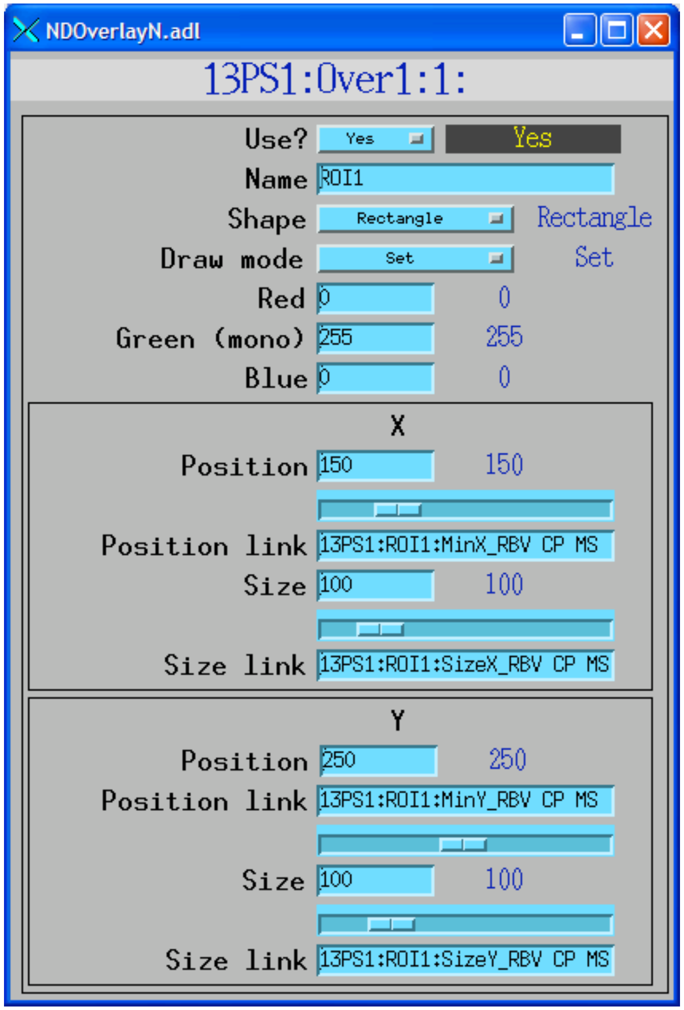








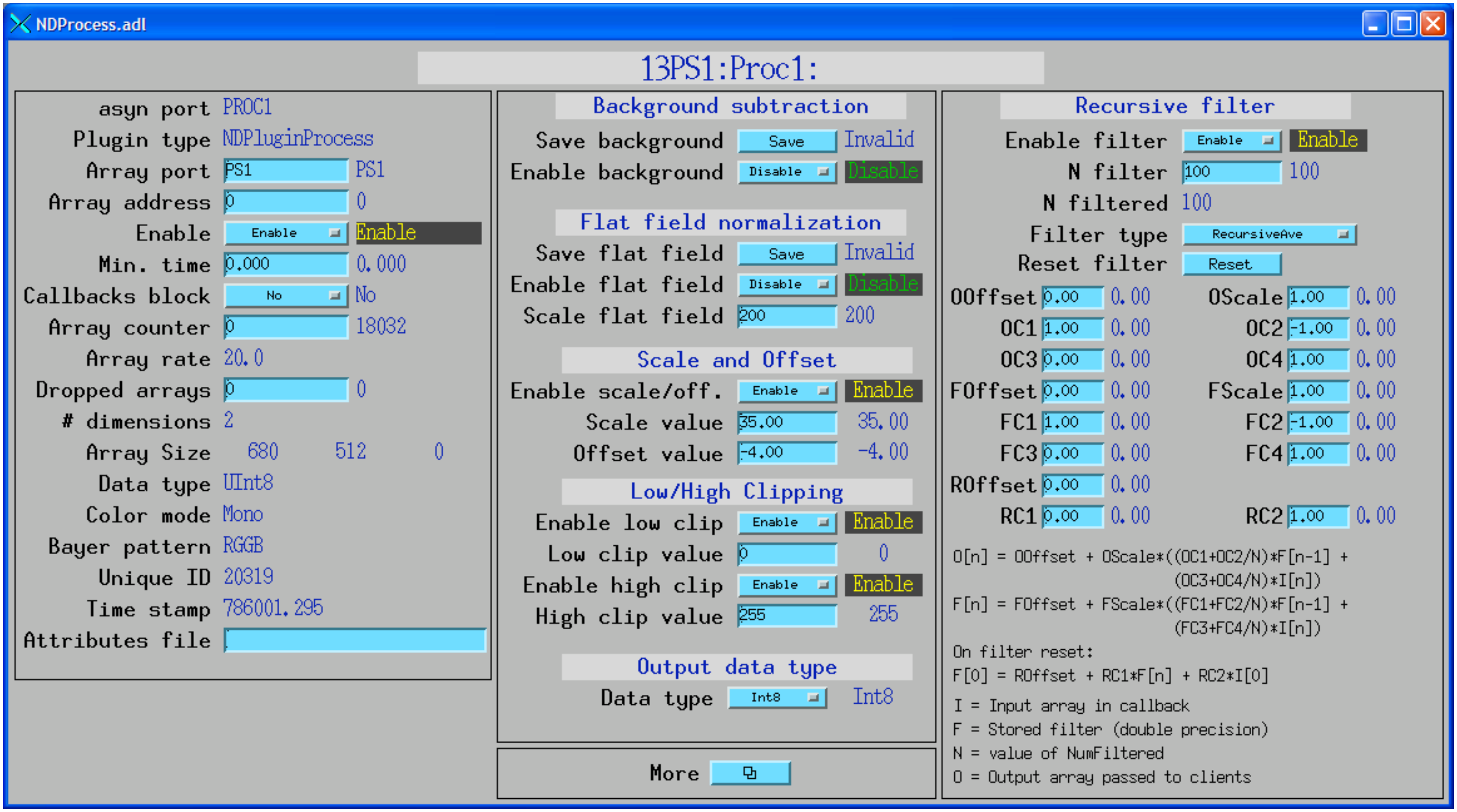
Overlay plugin



Centroid of laser pointer calculated by statistics plugin

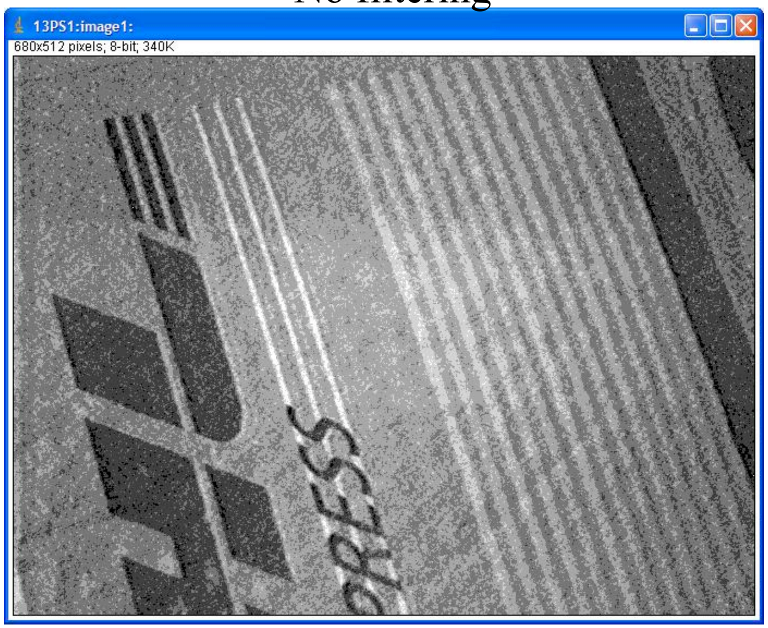
Cursor overlay X, Y position linked to centroid

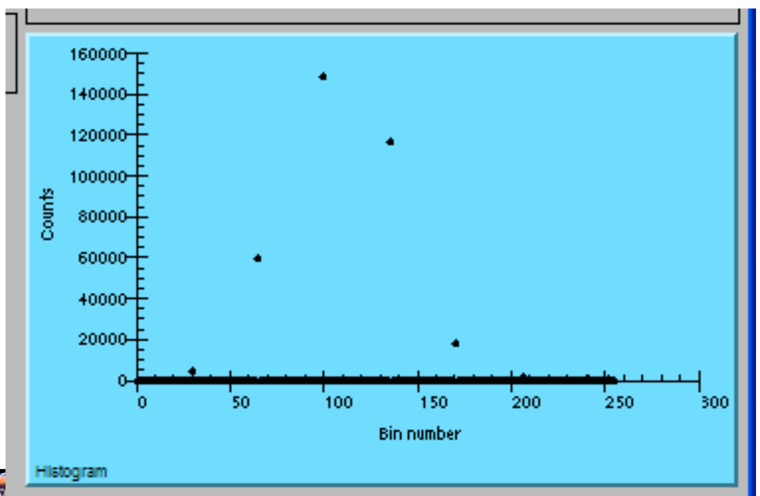
Processing plugin



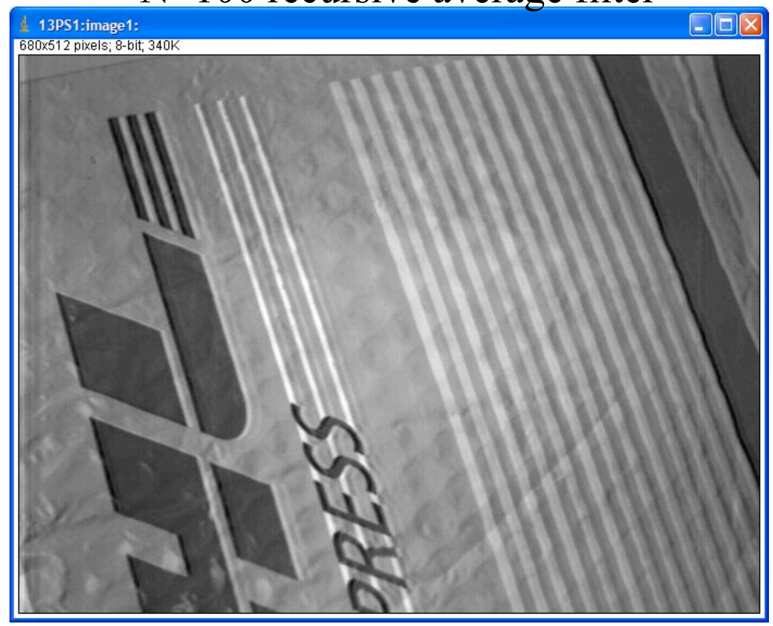
Processing plugin
30 microsec exposure time

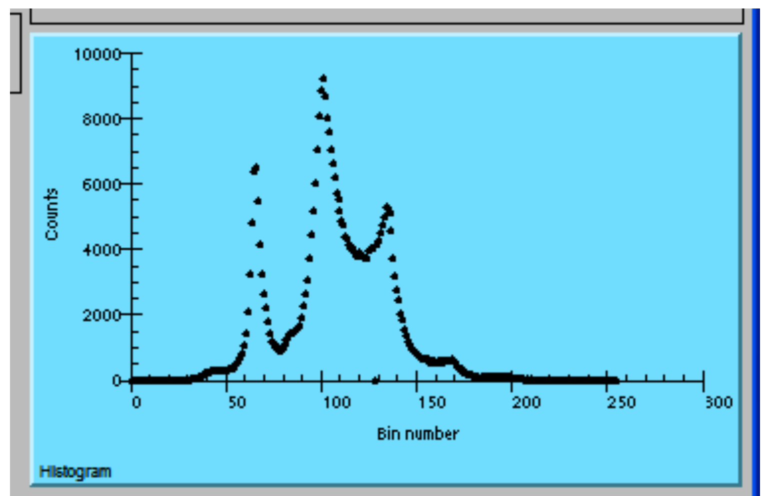
No filtering





N=100 recursive average filter





XIA Fast DSP Electronics for X-ray Fluorescence Detectors

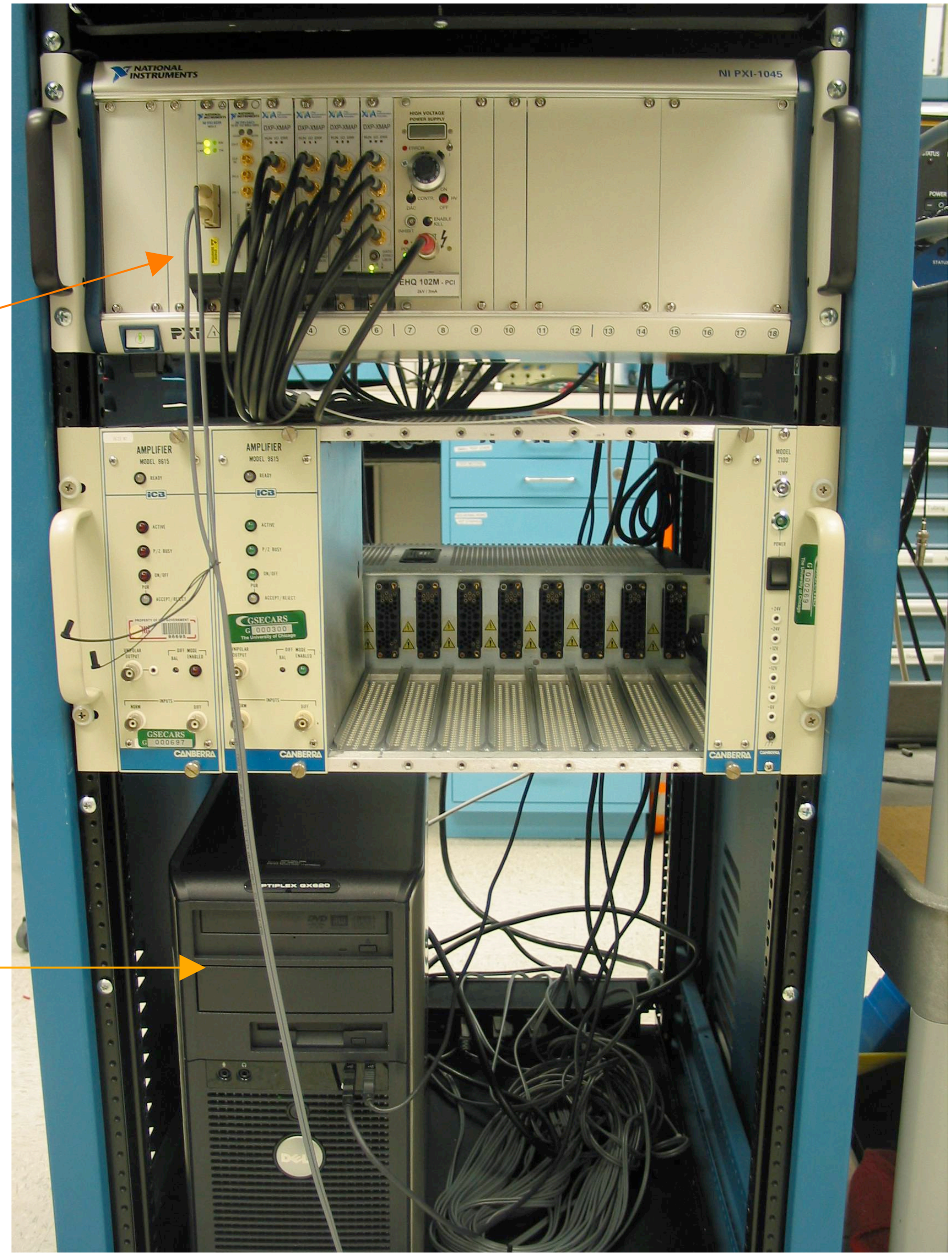
- Need a cost-effective way to collect XRF spectra from multi-element detector arrays
- Modern detectors, particularly silicon drift diodes (SDD) can run at >250,000 cps per detector, or >1,000,000 cps for a 4-element array like the quad Vortex
- Depending on the application, can thus get a usable signal (1,000 counts) in 1 ms.
  – Need to keep the overhead less than that!

xMAP electronics

- 4 channels per module
- 4 MB of memory per module. Used to buffer spectra or ROIs for very data collection
- Double-buffered to support simultaneous readout and acquisition
- 1 LEMO input for gate and trigger functions.
- Peaking times down to 125ns
- Supports both RC and reset preamps
- PXI/PCI interface which acheives ~30 MB/sec when reading out xMAP. More than 30 times faster than CAMAC.

xMAP

PXI crate with 4 xMAP units (16 channels) and fiber PXI to PCI interface



Windows control computer

xMAP Mapping Modes in Release 3-0

- MCA mapping
  - Spectra are buffered into onboard 4MB of memory
  - Double buffered for simultaneous readout and acquisition
  - With 2048 channel spectra each buffer holds 124 pixels maximum.
  - Performance: Limited by readout rate of xMAP over PXI/PCI, ~4,000 2048 channel spectra per second. For a 4-channel system (e.g. quad Vortex) this is 1,000 pixels/second. For a 100-element EXAFS detector it is 40 points/second
  - The first pixel in each buffer is sent to the MCA records for visual feedback on the data.
    - The buffer size can be decreased from 124 pixels when mapping slowly to get more rapid feedback.

xMAP Mapping Modes in Release 3-0

- ROI (SCA) mapping
  - Total counts in up to 16 ROIs per detector are collected into onboard 4MB of memory
  - Double buffered for simultaneous readout and acquisition
  - With 16 ROIs each buffer holds 5457 pixels maximum
  - Performance: Limited by xMAP overhead in pixel advance to about 100 microseconds/pixel, i.e. 10,000 pixels/second.
  - For a 16-element detector with 16 ROIs/detector this is 2.5M ROIs/second.

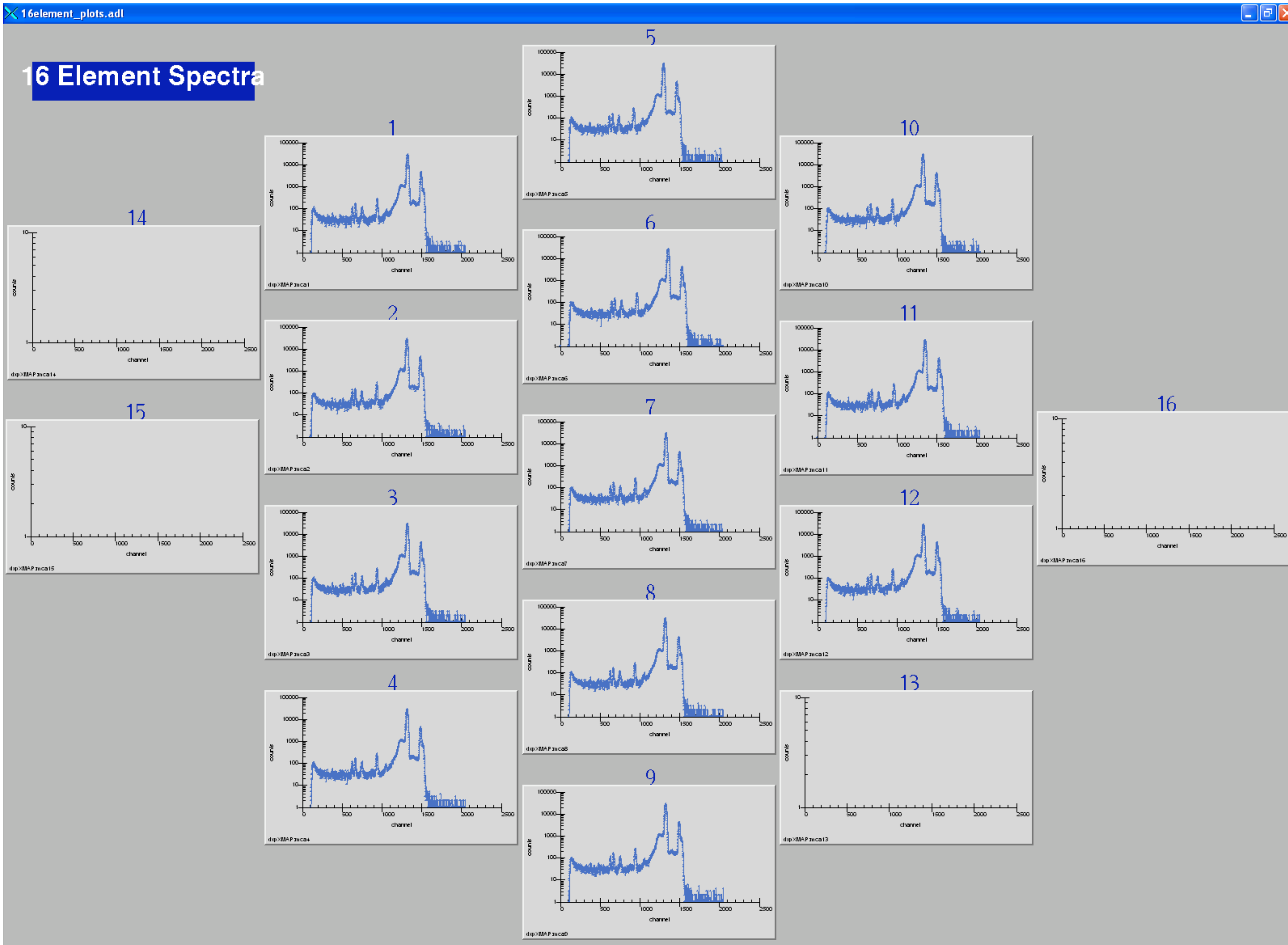
xMAP Mapping Modes in Release 3-0

- Pixel advance sources:
  - Software: This is a PV that can be written to at any time
  - External trigger: Trigger input to LEMO connector.
  - External sync: Like external trigger, but with option to divide input by N. Can be used to divide stepper motor pulses, for example, to have each pixel be 25 motor steps.

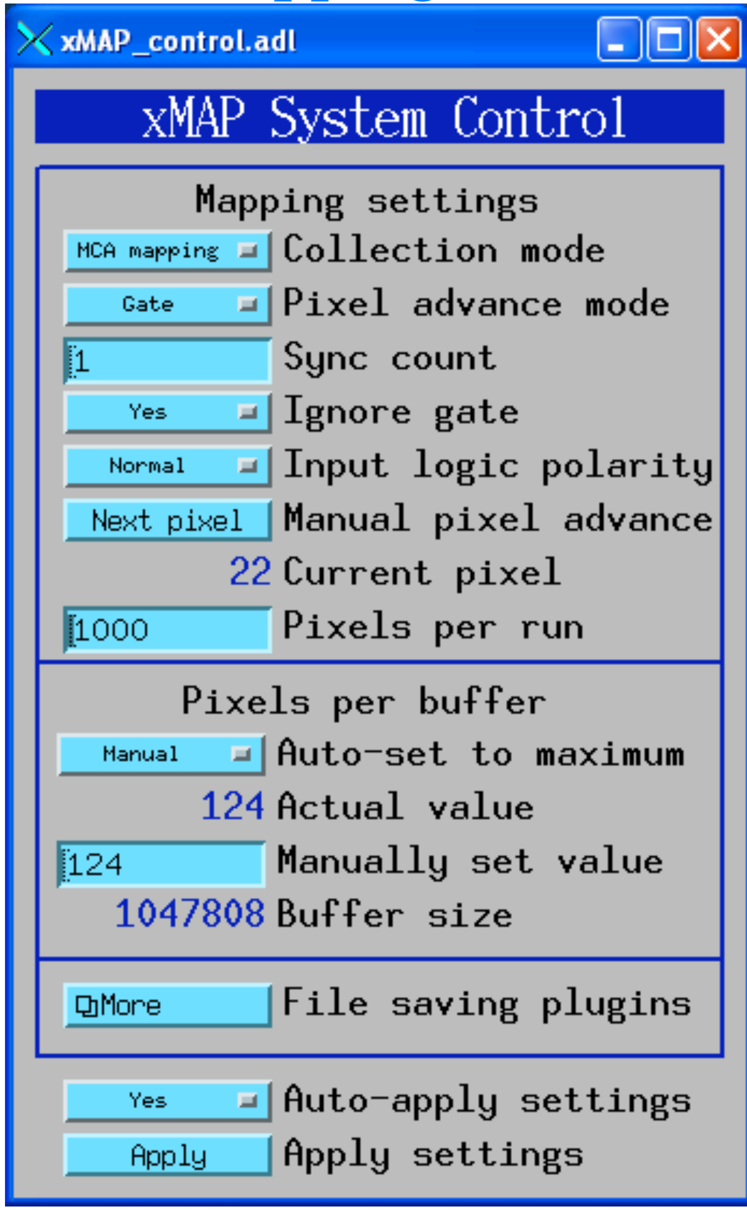
xMAP Mapping Modes in Release 3-0

- Data acquisition
  - When buffer fills up the EPICS software automatically reads it out and calls any NDArray plugins (from the areaDetector module) that have registered for callbacks.
  - The data are 16-bit 2-D arrays, 1047808 x N_modules.
    - The data in each array is a buffer containing the spectral data, as well as live time, real time, input counts and output counts.
  - The plugins will normally be file-saving plugins. The netCDF, TIFF and NeXus/HDF plugins from areaDetector can all be directly used. The JPEG plugin will not be useful!
  - The netCDF plugin can stream data continuously to a single netCDF file. The TIFF plugin writes each 2-D array to a separate TIFF file
  - IDL and Python routines are available to extract the data from the netCDF files.
  - Continuously streaming data at the rates on the previous slide

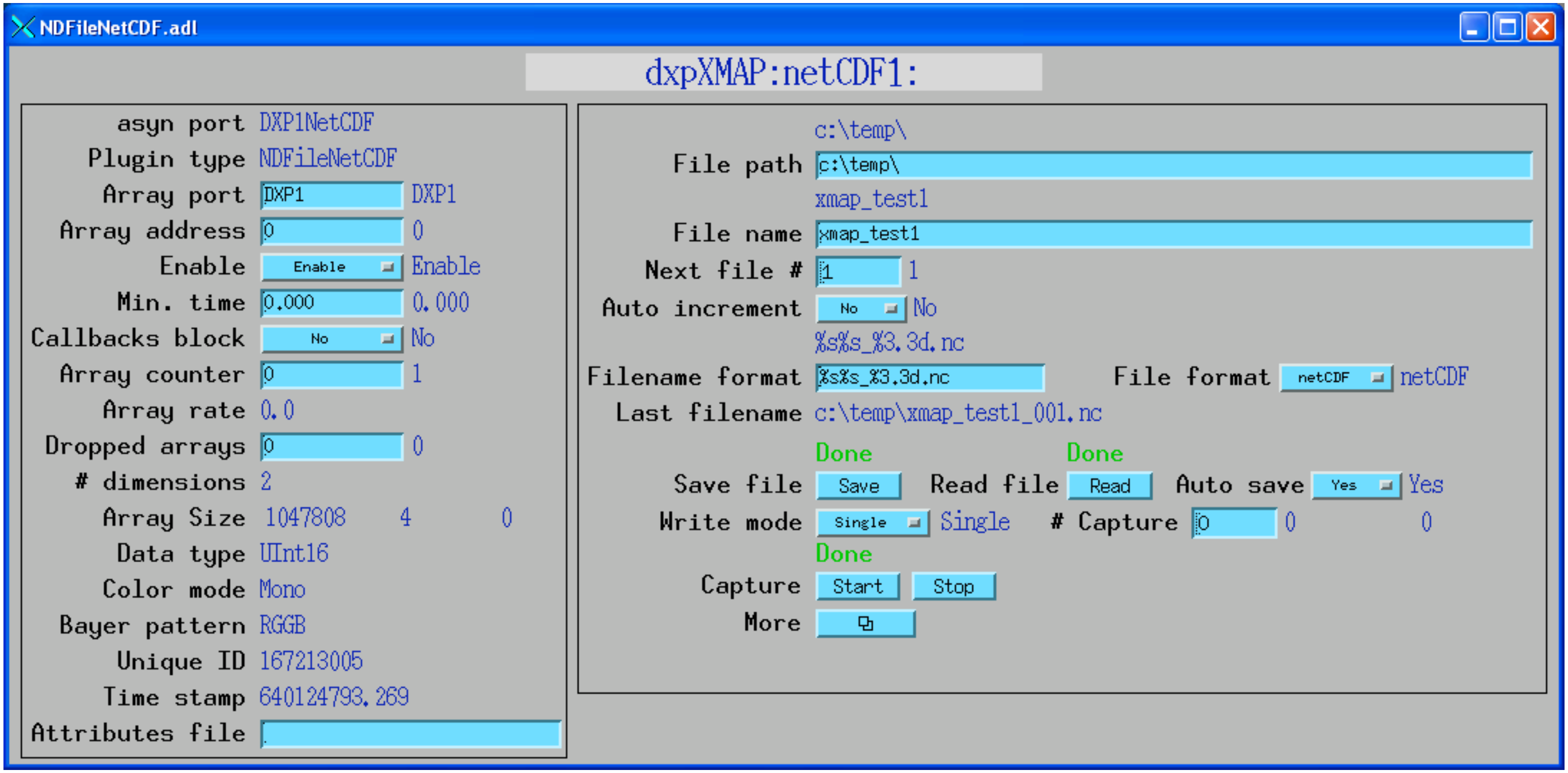
16 element combined spectra



xMAP mapping mode setup



netCDF file saving plugin for mapping modes

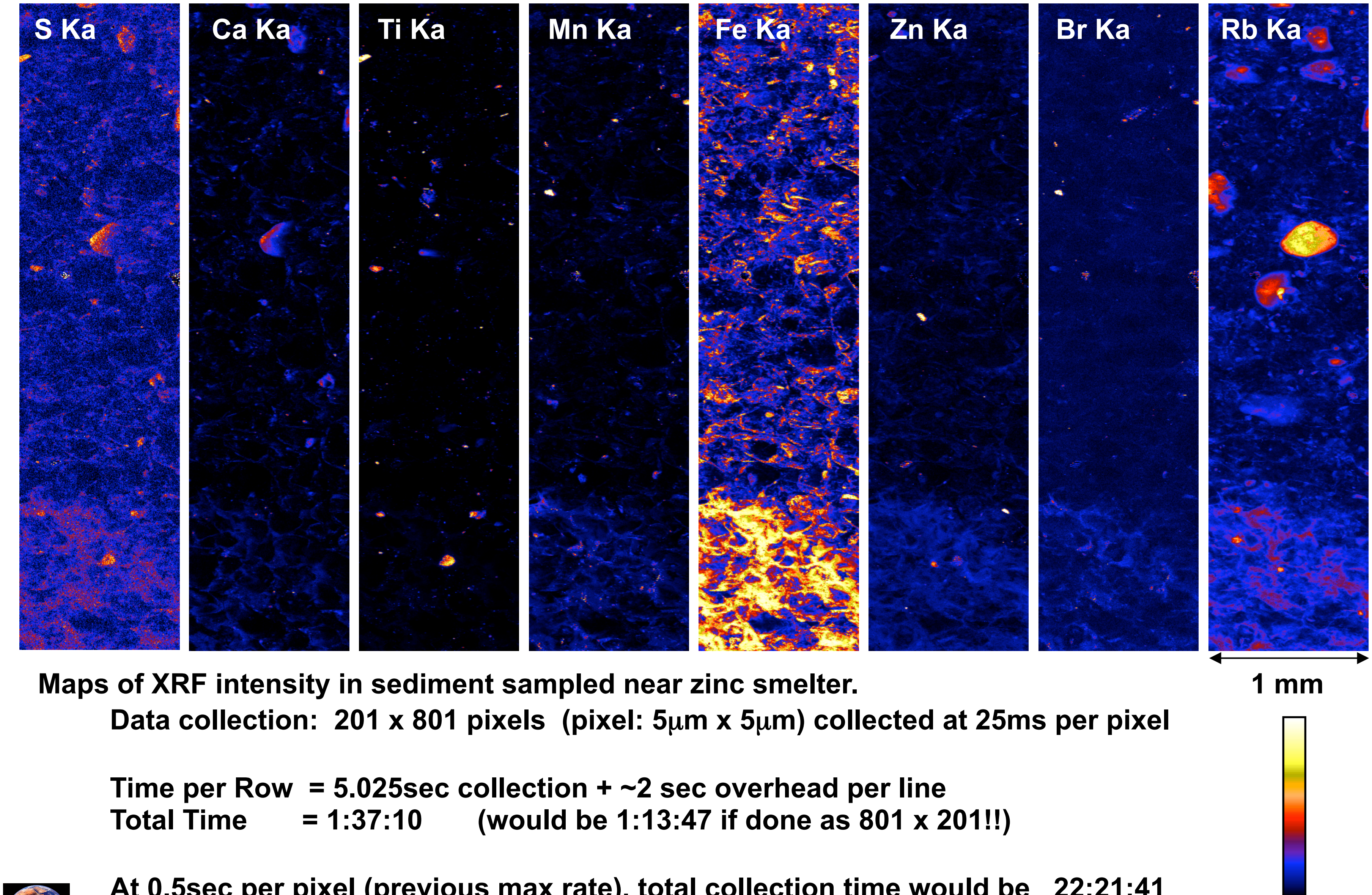


First Results with xMAP MCA Mapping Mode
Matt Newville, 13-ID-C

- SII quad Vortex detector
- Sample stage driven with Newport XPS motor controller running trajectory scanning software, continuous stage motion
- Bi-directional stage motion
- XPS puts out a trigger pulse at each pixel
- XPS captures actual stage position when each trigger pulse is output
- Trigger pulse goes to channel advance on SIS multichannel scaler to capture I0 from ion chamber & V/F converter
- SIS output pulse triggers xMAP trigger input
- Current version of software collects 1 row of image in xMAP buffer and writes to netCDF file
  - Could do an entire image into a single file to lower overhead.
  - Need to see if another process can read the file for display update
- Python software reads file, converts to an older format that can be displayed by Matt's Python collection software.
  - Adds additional overhead, but will be replaced with a new system Matt is designing

XRF Fast Mapping Mode example 1

G. Morin, F. Juillot Univ Paris VI



Maps of XRF intensity in sediment sampled near zinc smelter.

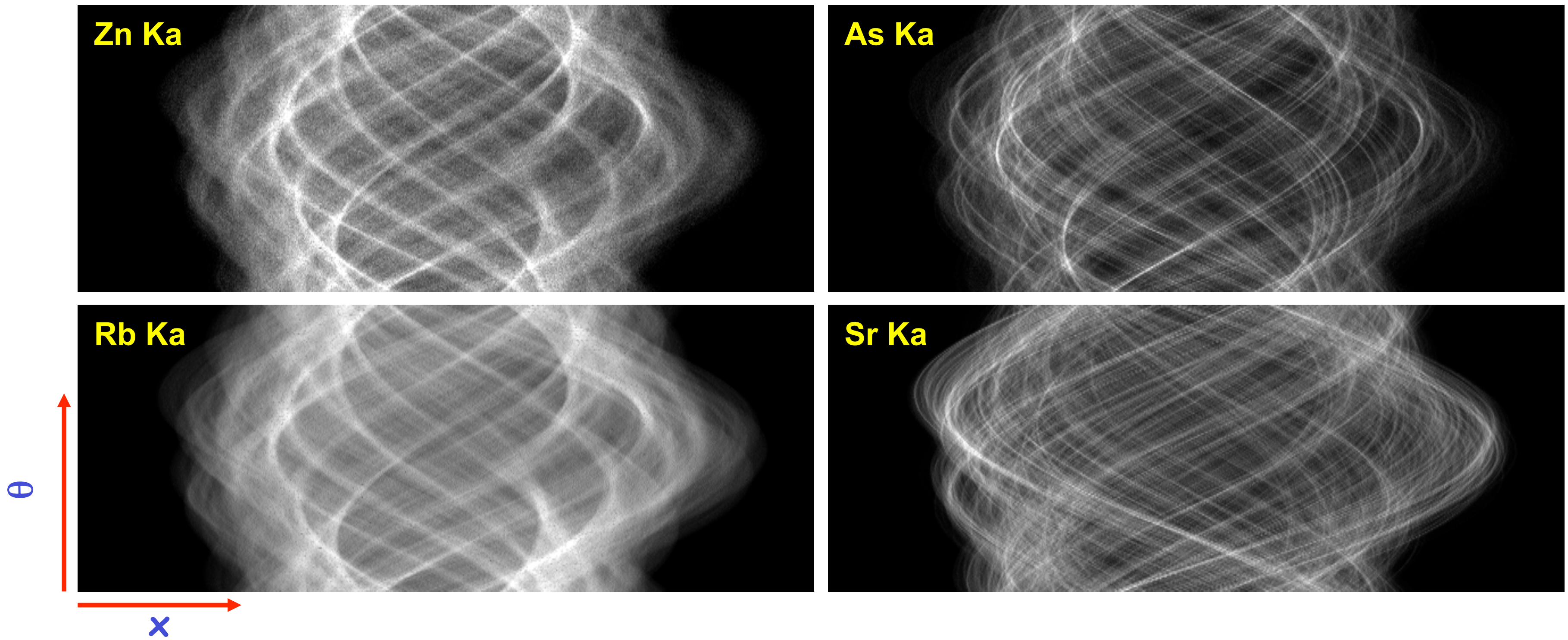
Data collection: 201 x 801 pixels (pixel: 5µm x 5µm) collected at 25ms per pixel

Time per Row = 5.025sec collection + ~2 sec overhead per line
Total Time = 1:37:10 (would be 1:13:47 if done as 801 x 201!!)

At 0.5sec per pixel (previous max rate), total collection time would be 22:21:41

XRF Fast Mapping Mode example 2: Fluorescence Tomography

Anne-Marie Carey, U. of Aberdeen, Kirk Scheckel US-EPA:
Distribution of Heavy Metals, especially As, in Rice



X-θ maps of XRF intensity in panicle (small stem to grain) in rice, grown in As(III)-spiked solution

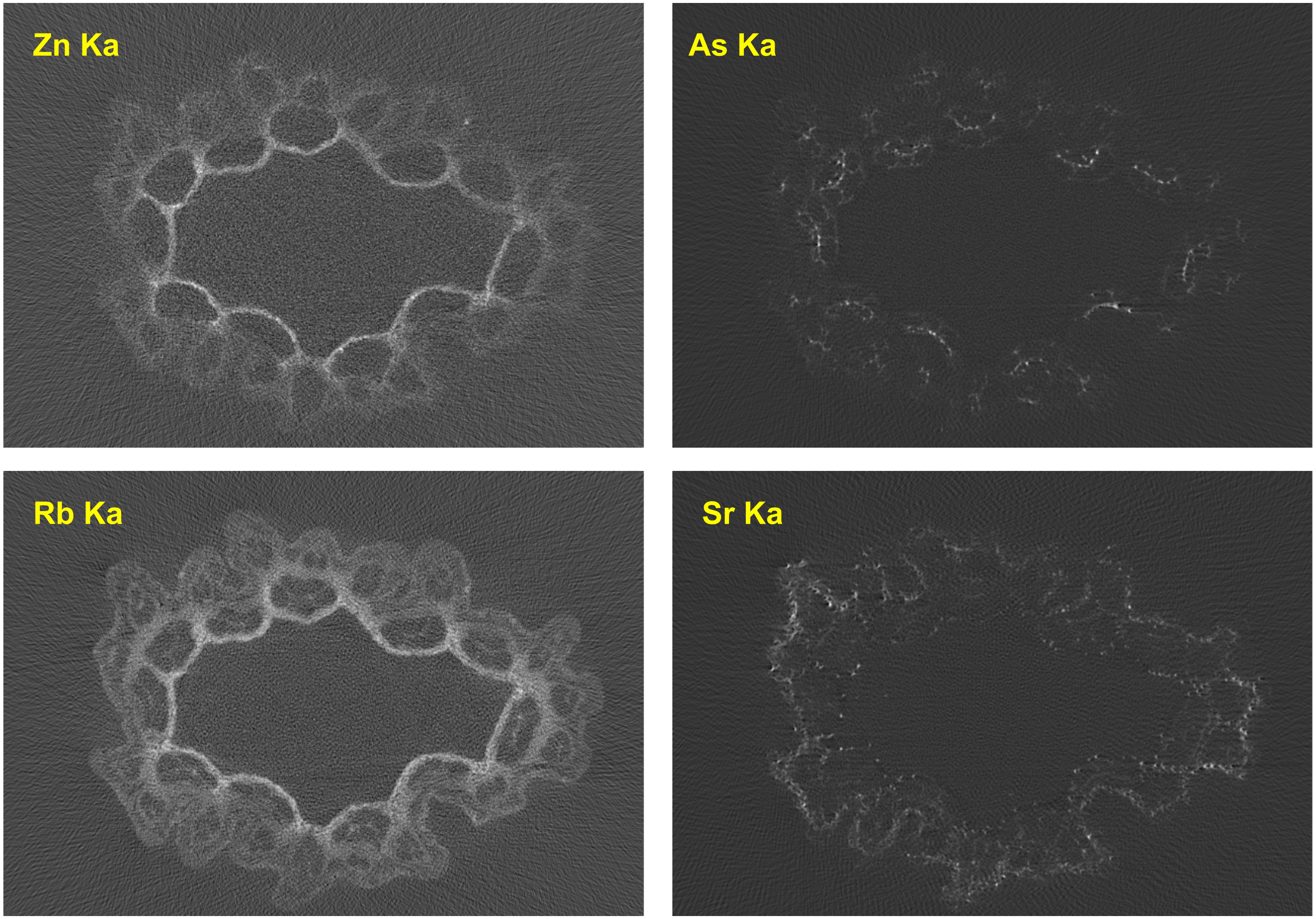
Data collection: 648 x 181 pixels (pixel: 2μm x 1degree) collected at 30ms per pixel

Time per Row = 20.5sec collection + ~2 sec overhead per line
Total Time = 1:07:20

At 0.5sec per pixel, total collection time would be 17:11:42

XRF Fast Mapping Mode example 2: Reconstructed Slices

Anne-Marie Carey, U. of Aberdeen, Kirk Scheckel US-EPA



Rb: marks phloem transport

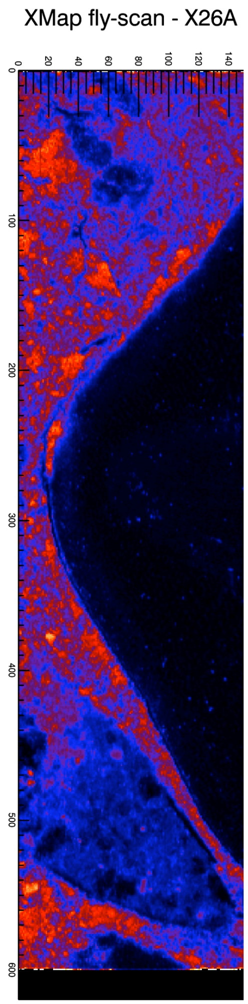
Sr: marks xylem transport

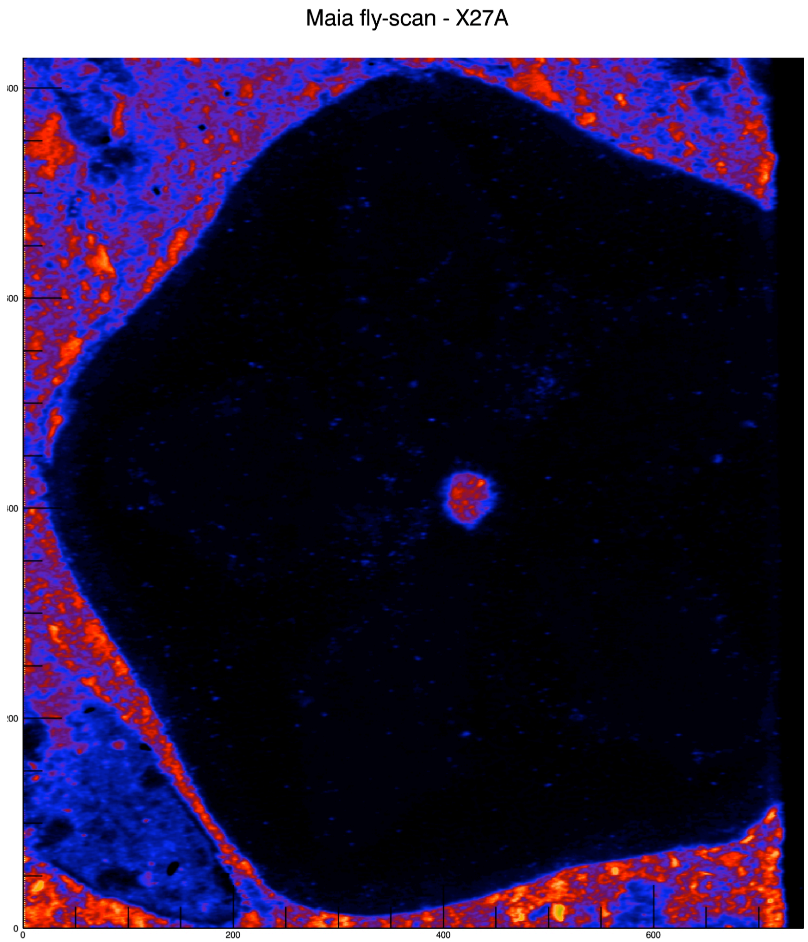
Comparison of 11-element xMAP vs 96-element Maia detector
Tony Lanzirotti, Univ. of Chicago X-26A NSLS

- Maia system
  - 96-element prototype
  - Beamline X-27A at NSLS
  - 7 micron pixel size
  - 138 micron/sec scan speed
  - = 38 ms/pixel dwell time
  - 743x829 pixels
  - 6.3 hours
- xMAP system
  - First tests last weekend on X-26A beamline
  - 9 element Ge detector plus 2 Vortex detectors = 11 detector elements
  - Sample stage driven with OMS MAXv controller
  - Stepper pulses drive channel advance on SIS multichannel scaler to capture I0 from ion chamber & V/F converter
  - Uni-directional stage motion
  - SIS output pulse triggers xMAP trigger input
  - Software collects 1 row of image in xMAP buffer and writes to netCDF file
  - Entire 2-D scan done with 20 line IDL script
  - Same pixel size and scan speed as with Maia detector
  - 690x150 pixels

Comparison of 11 element xMAP vs 96-element Maia detector

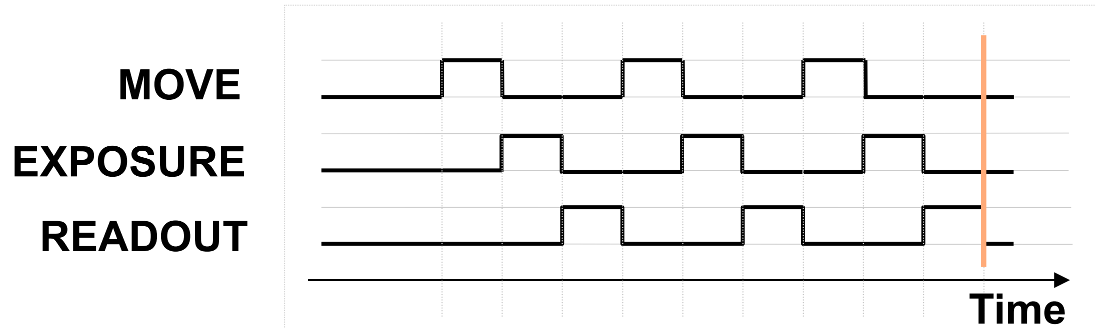
- Echinoderm sample
- Fe Kα map
  - 7 micron pixel size
  - 138 micron/sec scan speed
  - = 38 ms/pixel dwell time



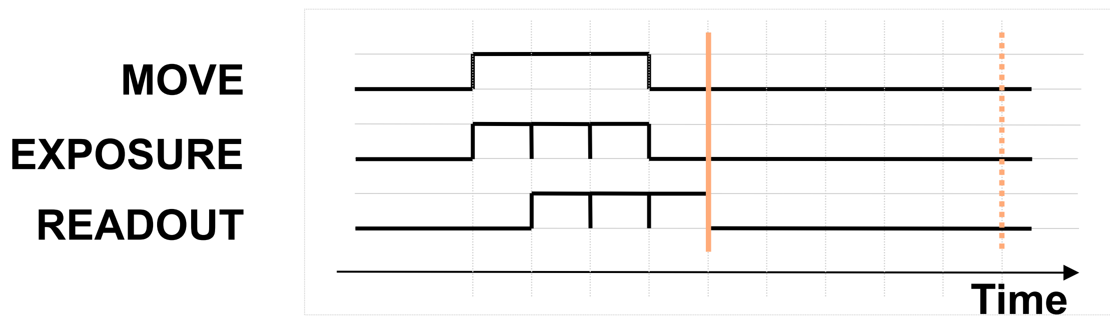


ESRF
Continuous scans

- *From Sequential*



- *To Parallel*



- *Practical Issues*
  - Distributed System
  - Multiple detectors
  - Short exposure time

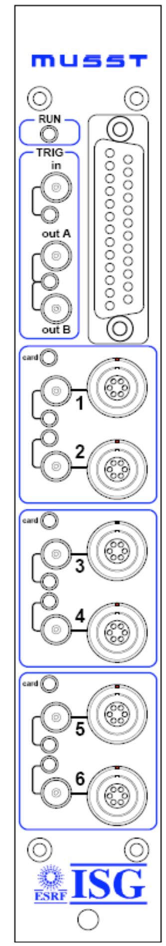
- Synchronization
- Data Buffering

ESRF
Emanuel Papillon
Continuous scans : synchronization

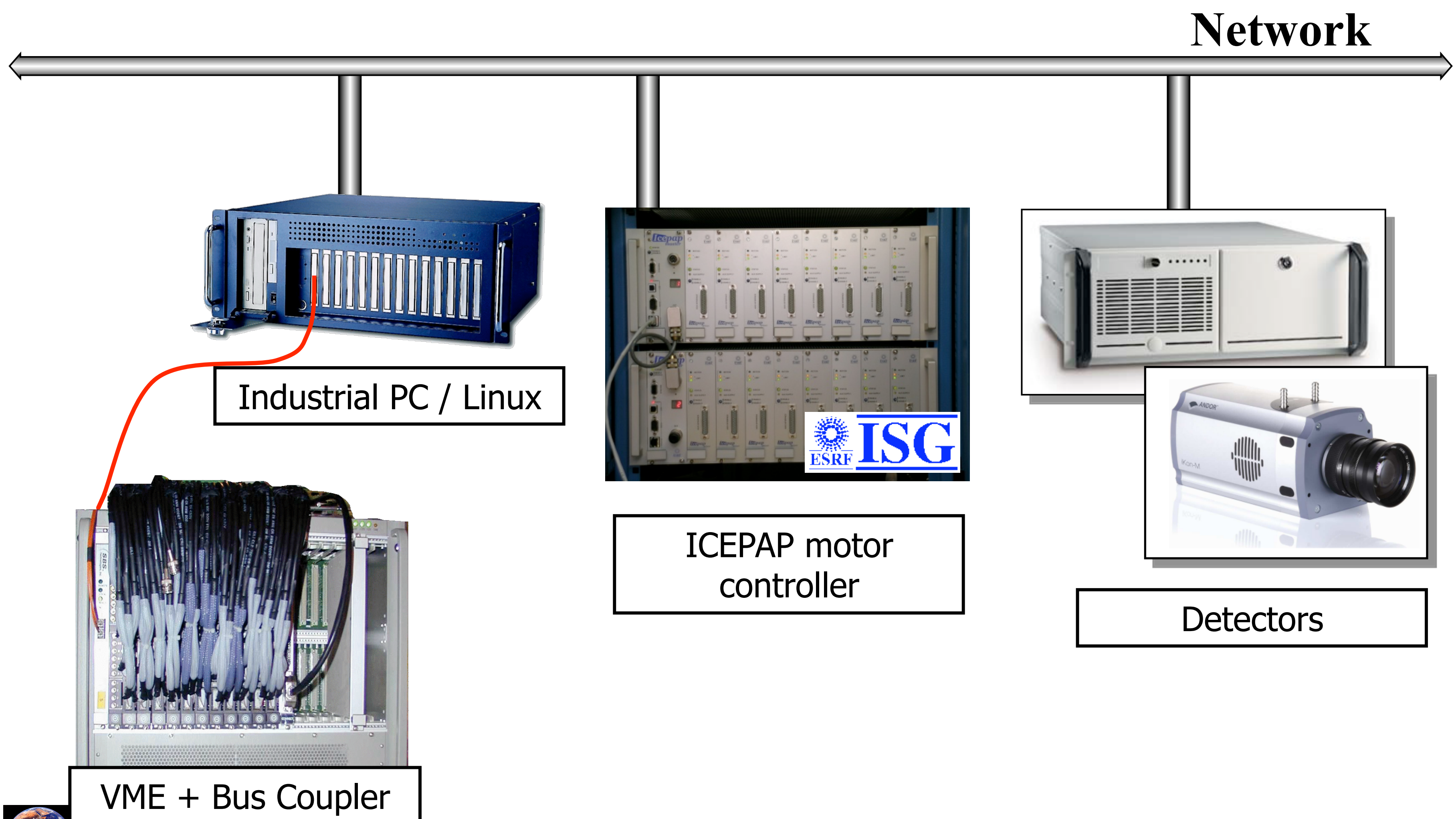
– *By software* :
  - Synchro. with time: no wiring, good on single host
– *By hardware* :
  - Needs wiring, good for distributed hosts
  - Synchro. with time : **P201**, **OPIOM** boards
  - Need more : motor position, mixed time/position, …

MUSST board

– **Very flexible event generator**
– **Hardware buffering**
– **Input signals (6):**
  - Digital (level and counter)
  - Analog
  - Positioning (motor steps and encoders)
– **Host Interface :** GPIB



ESRF
Hardware overview

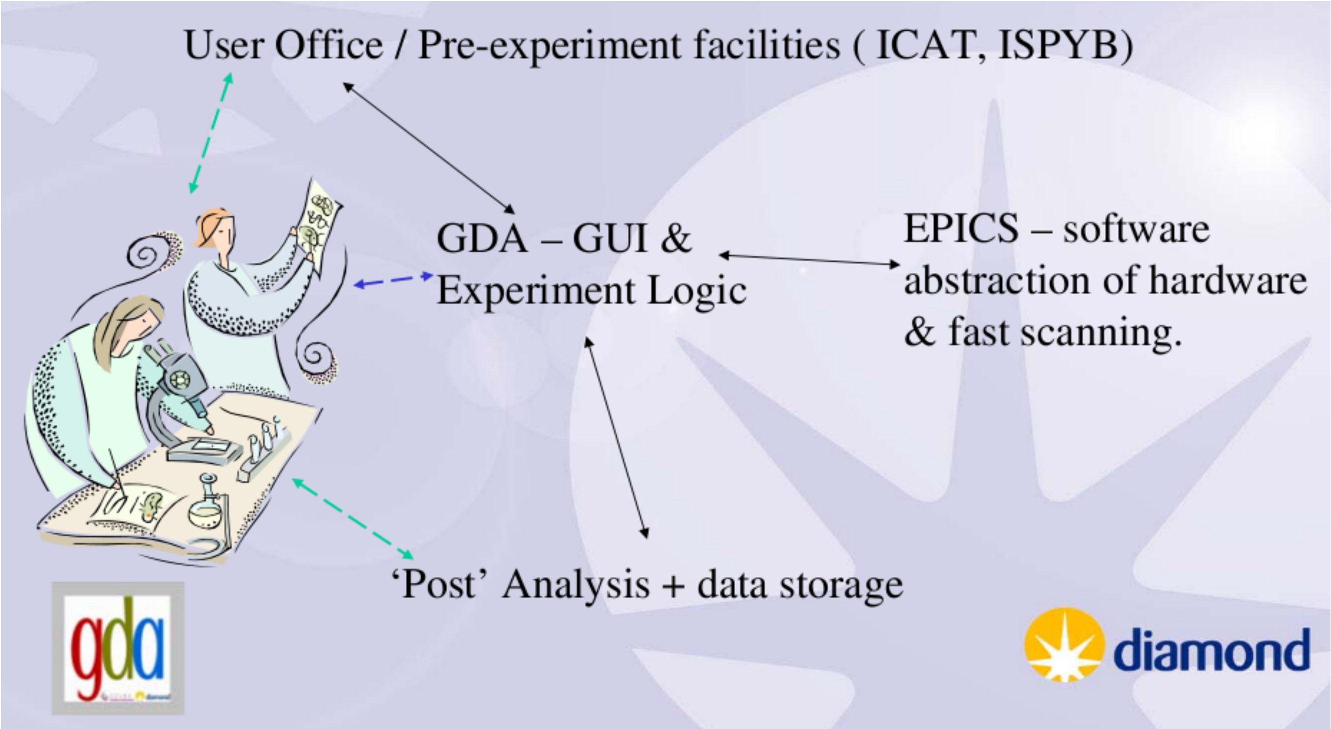


User Interfaces at GSECARS (for example)

- SPEC
  - Used to control 2 large Newport diffractometers
  - Only software we have that understands reciprocal space and the geometry of the diffractometers
  - It is very familiar to the scattering users
  - But they understand its limitations as a programming language
- Tomography
  - IDL GUI that talks directly to CCD detector and to EPICS motors, etc.
  - Will replace CCD interface with areaDetector
  - Simple, easy for users to learn
  - IDL program acts as a server via EPICS PVs, so scripts on other computers can order it to collect a data set while they change energy, temperature, etc.
- XRF microprobe and EXAFS
  - Python GUI and data collection code by Matt Newville
- No unifying user interface beyond medm
  - Medm is very dated. There are EPICS replacements on the way
    - Control System Studio (CSS) Synoptic Display System (SDS)
    - BOY
- What we'd like:
  - A common GUI and common scripting language that can do all of the above and more
  - GDA?

Diamond Data Acquisition System – GDA

- EPICS provides most hardware interface and low level experiment synchronisation
- GDA integrates the EPICS system and other devices to give the user level functionality



What is GDA

- One or more ObjectServers each holding collection of Java objects. The objects may themselves provided a connection to external hardware controllers e.g. EPICS devices
- A mechanism for one object to access another across ObjectServers
- A mechanism for objects to notify other objects of changes of state
- One ObjectServer contains a Jython Interpreter This has access to the other objects in the system
- A highly flexible Scan mechanism that can be used to a wide range of measurements
- Rich GUI clients that also have access to the other objects in the system, including the Jython Interpreter The Rich GUI is constructed using Eclipse plugin framework It also has an ObjectServer embedded in it
- Eclipse Plugins exist to give access to the objects, jython interpreter and provide data visualisation

What is GDA

- Very powerful common scan mechanism that can easily be used to scan any combination of real or virtual devices at any level of nesting

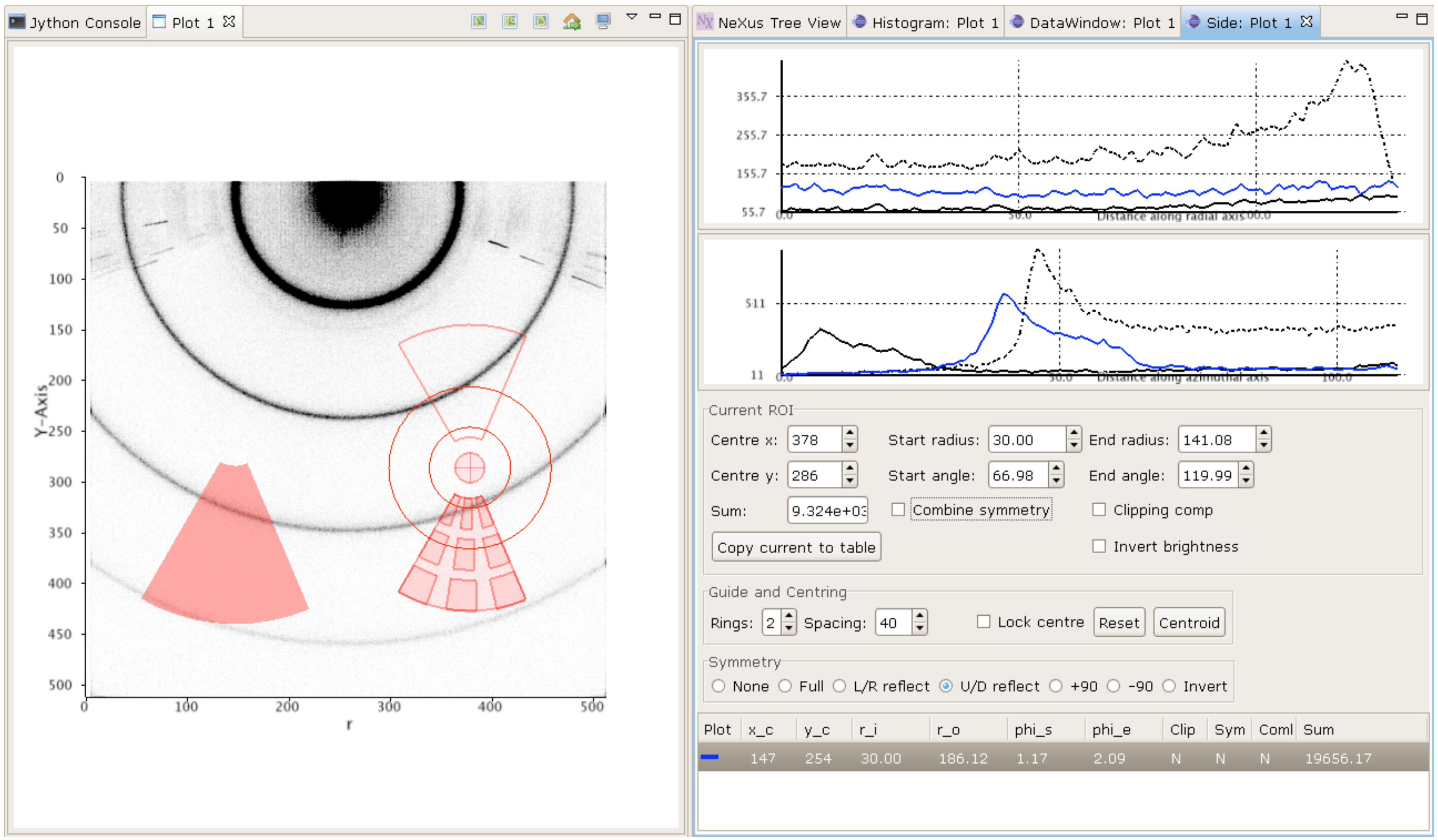
- Scan mechanism creates plottable data and common format data files that can cope with any level of complexity of data. But the pipeline for handling scan results can be extended and adapted easily

- Scanned devices can be written in Java or Jython, by engineers and beamline users. They can be as simple as returning the beam current or as complex as configuring and reading a multi element Germanium detector , or driving a combined motor trajectory scan with synchronised data measurement

- A scan of energy keeping hkl fixed at [1 0 1] , with polarisation set to 90, at each energy reading the value from a counter timer exposed for 1 second is read, plotted and written to file:

```
scan energy 5.95 6.05 .0005 hkl [1 0 1] pol 90 ct 1
```

GDA


Jython Console
Plot 1
NeXus Tree View
Histogram: Plot 1
DataWindow: Plot 1
Side: Plot 1
Y-Axis
r
Distance along radial axis
Distance along azimuthal axis
Current ROI
Centre x: 378
Centre y: 286
Sum: 9.324e+03
Start radius: 30.00
Start angle: 66.98
End radius: 141.08
End angle: 119.99
Combine symmetry
Clipping comp
Invert brightness
Copy current to table
Guide and Centring
Rings: 2
Spacing: 40
Lock centre
Reset
Centroid
Symmetry
None
Full
L/R reflect
U/D reflect
+90
-90
Invert
Plot x_c y_c r_i r_o phi_s phi_e Clip Sym Coml Sum
147 254 30.00 186.12 1.17 2.09 N N N 19656.17

GDA for APS?

- We should investigate GDA as a unifying framework for user interfaces at APS
- Ken Evans has been getting it running here
- It is being seriously considered for NSLS-II
- ESRF is also investigating it for their future “beyond SPEC”

Summary

- EPICS gives us excellent tools for low-level high-performance data acquisition
- New software within this framework is needed to make the "integrated beamline" a reality
- New hardware is needed to move to the next level
    - Coordinating undulator scans, motion in multiple controllers, etc.
    - Data acquisition with non-pulse sources
- We really need better and more uniform user interfaces, and thus they need to be easy to develop
    - GDA is worth a hard look, particularly if NSLS-II and/or ESRF adopt it